UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/805,963	03/22/2004	Kent F. Hayes JR.	RSW920030236US1	2602

45541 7590 09/19/2008

HOFFMAN WARNICK LLC
75 STATE ST
14TH FLOOR
ALBANY, NY 12207

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
09/19/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOCommunications@hoffmanwarnick.com

Office Action Summary	Application No.	Applicant(s)	
	10/805,963	HAYES, KENT F.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *June 16, 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated June 16, 2008, responding to the Office action mailed April 7, 2008 provided in the rejection of claims 1-40.

Claims 1-40 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are not persuasive. Please see the section of "Response to Arguments" for details.

2. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-7, 9, 11-17, 19-26, 28-29, and 31-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Klicnik et al. (Pub. No. US 2002/0184226 A1) (hereinafter 'Klicnik') in view of Liang et al. (*Bundle Dependency in Open Services Gateway Initiative Framework Initialization, 2002, IEEE*) (hereinafter 'Liang') and further in view of Clohessy et al., (Pub. No. US 2003/0023661 A1) (hereinafter 'Clohessy')

4. **As to claim 1** (Previously Presented), Klicnik discloses a computer-implemented method for resolving prerequisites for native applications comprising:

- packaging a native application for a client device and corresponding dependency information within a first OSGi bundle on a server (e.g., [0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the corresponding dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]);

Although Klicnik discloses OSGI bundles ([0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of *Bundle Dependency in Open Services Gateway Initiative Framework Initialization*, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (e.g., Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (e.g., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16)

Klicnik and Liang do not explicitly disclose polling the client device by the server to determine if the client device has the at least one other prerequisite; obtaining the at

least one prerequisite if the client device does not have the at least one prerequisite; and loading the at least one prerequisite and the native application on the client device.

However, in an analogous art of *Runtime-Resource Management*, Clohessy discloses:

- polling the client device by the server to determine if the client device has the at least one other prerequisite (e.g., Fig. 4 – depicting loading one or more new application component into a portable device - it shows the recursive path used to resolve prerequisites, steps 104 → 106 → 108 → 109 → 110 → 112 →104 etc. until step-114 or END reached; [0038], Lines 12-15 – Alternatively, **a server** could **identify** one or more new application components required to upgrade/update an application component already stored in **the portable device** (client device); [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed – emphasis added);
- obtaining the at least one prerequisite if the client device does not have the at least one prerequisite (e.g., [0041] – the client device determining its available resources, and these are communicated to the server since; [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed); and
- loading the at least one prerequisite (e.g., [0038], Lines 12-15 - … one or more new application component required to upgrade/update an application component …)

and the native application (e.g., [0035], Lines 7-10 – Preferably, each RDL is stored with its associated application component in a single file such as JAVA™ Archive (JAR) file or J9 executable (JXE) file that can be run by the IBM J9 VM) on the client device (e.g., [0046] – In the preferred embodiment, the new application component is **downloaded from a network server to the portable device** in a know manner).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Clohessy into the Klicnik-Liang's system to further provide polling the client device by the server to determine if the client device has the at least one other prerequisite; obtaining the at least one prerequisite if the client device does not have the at least one prerequisite; and loading the at least one prerequisite and the native application on the client device in Klicnik-Liang's system.

The motivation is that it would further enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Clohessy's system which offers significant advantages for improved resource management methods, systems, and products which protect runtime system resources from poorly designed or destructive application components as once suggested by Clohessy (e.g., [0006])

5. **As to claim 2** (Original) (incorporating the rejection in claim 1), Liang discloses the method, further comprising registering the packaged native application and first OSGi bundle after the packaging step, wherein the registering step comprises storing the corresponding dependency information (e.g., Sec. 1, 1st Para., Lines 12-14 – open

services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12)

6. **As to claim 3** (Original) (incorporating the rejection in claim 1), Klicnik discloses the method, wherein the polling step comprises: identifying the at least one prerequisite to the client device (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3); and Liang discloses receiving a response from the client device, wherein the response indicates whether the client device has the at least one prerequisite (e.g., Sec. of b) Modify the Bundle Management Strategy of the Framework, Lines 1-4)

7. **As to claim 4** (incorporating the rejection in claim 1) (Original), Klicnik discloses the method, further comprising: determining the at least one prerequisite, prior to the packaging step; and generating the corresponding dependency information based on the at least one prerequisite (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

8. **As to claim 5** (Original) (incorporating the rejection in claim 1), Klicnik discloses the method, wherein the at least one prerequisite comprises another native application (e.g., [0018], Lines 5-13)

9. **As to claim 6** (Original) (incorporating the rejection in claim 1), Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within a second OSGi bundle, and wherein the obtaining step comprises obtaining the second OSGi bundle (e.g., Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.)

10. **As to claim 7** (Original) (incorporating the rejection in claim 6), Liang discloses the method, wherein loading step comprises:

- installing the first OSGi bundle and the second OSGi bundle within an OSGi environment of the client device (e.g., Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.);
- deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

11. **As to claims 9** (Original) (incorporating the rejection in claim 1), Liang discloses the method and the program product, wherein the dependency information is expressed

as a package import statement (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para., Lines 5-11 – import-package field in its manifest file)

12. **As to claim 11** (Previously Presented), Klicnik discloses a computer-implemented method for resolving prerequisites for native applications, comprising:

- packaging a native application for a client device and corresponding dependency information within a first bundle on a server (e.g., [0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (e.g., Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032])

Although Klicnik discloses OSGI bundles (e.g., [0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of *Bundle Dependency in Open Services Gateway Initiative Framework Initialization*, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (e.g., Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para.,

Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (e.g., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16)

Klicnik and Liang do not explicitly disclose polling the client device to determine if the client device has the at least one other prerequisite; obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and installing the first bundle and the second bundle within an environment of the client device.

However, in an analogous art of *Runtime-Resource Management*, Clohessy discloses:

- polling the client device to determine if the client device has the at least one other prerequisite (e.g., Fig. 4 – depicting loading one or more new application component into a portable device - it shows the recursive path

used to resolve prerequisites, steps 104 → 106 → 108 → 109 → 110 → 112 →104 etc. until step-114 or END reached; [0038], Lines 12-15 – Alternatively, **a server** could **identify** one or more new application components required to upgrade/update an application component already stored in **the portable device** (client device); [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed – emphasis added);

- obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite (e.g., [0038], Lines 12-15 - ... one or more new application component required to upgrade/update an application component ...) is packaged within a second bundle (e.g., [0035], Lines 7-10 – Preferably, each RDL is stored with its associated application component in a single file such as JAVA™ Archive (JAR) file or J9 executable (JXE) file that can be run by the IBM J9 VM) that is accessible to the server (e.g., [0041] – the client device determining its available resources, and these are communicated to the server since; [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed); and

- installing the first bundle and the second bundle within an environment of the client device (e.g., Fig. 4, elements 108 and 114 show loading the final set of OSGi bundles on the client device)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Clohessy into the Klicnik-Liang's system to further provide polling the client device to determine if the client device has the at least one other prerequisite; obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and installing the first bundle and the second bundle within an environment of the client device in Klicnik-Liang's system.

The motivation is that it would further enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Clohessy's system which offers significant advantages for improved resource management methods, systems, and products which protect runtime system resources from poorly designed or destructive application components as once suggested by Clohessy (e.g., [0006])

13. **As to claim 12** (Original) (incorporating the rejection in claim 11), Liang discloses the method, wherein the first OSGi bundle and the second OSGi bundle are registered on the server after being packaged with the first native application and the at least one prerequisite (e.g., Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service

security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12)

14. **As to claims 13-15**, please refer to claims **3-5** as set for the above accordingly.

15. **As to claim 16** (Original) (incorporating the rejection in claim 11), Liang discloses the method, further comprising:

- deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

16. **As to claim 17** (Original) (incorporating the rejection in claim 11), Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (e.g., Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.)

17. **As to claim 19** (Previously Presented), Klicnik discloses a computerized system for resolving prerequisites for native applications, comprising:

- a packaging system for packaging a native application for a client device and corresponding dependency information within a first bundle on a server (e.g., [0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (e.g., Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]);

Although Klicnik discloses OSGI bundles (e.g., [0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of *Bundle Dependency in Open Services Gateway Initiative Framework Initialization*, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (e.g., Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 – bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (e.g., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16)

Further, Klicnik and Liang do not explicitly disclose a communication system for polling the client device to determine if the client device has the at least one other prerequisite; a resolution system for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and a bundle loading system for loading the first bundle and the second bundle on the client device.

However, in an analogous art of *Runtime-Resource Management*, Clohessy discloses:

- a communication system for polling the client device to determine if the client device has the at least one other prerequisite (e.g., Fig. 4 – depicting loading one or more new application component into a portable device - it shows the recursive path used to resolve prerequisites, steps 104 → 106 → 108 → 109 → 110 → 112 →104 etc. until step-114 or END reached; [0038], Lines 12-15

– Alternatively, **a server** could **identify** one or more new application components required to upgrade/update an application component already stored in **the portable device** (client device); [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed – emphasis added);

- a resolution system for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server (e.g., [0041] – the client device determining its available resources, and these are communicated to the server since; [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed); and
- a bundle loading system for loading the first bundle and the second bundle on the client device (e.g., Fig. 4, elements 108 and 114 show loading the final set of OSGi bundles on the client device)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Clohessy into the Klicnik-Liang's system to further provide a communication system for polling the client device to determine if the client device has the at least one other prerequisite; a resolution system for obtaining the at least one prerequisite if the client device does not have the at least

one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and a bundle loading system for loading the first bundle and the second bundle on the client device in Klicnik-Liang's system.

The motivation is that it would further enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Clohessy's system which offers significant advantages for improved resource management methods, systems, and products which protect runtime system resources from poorly designed or destructive application components as once suggested by Clohessy (e.g., [0006])

18. **As to claim 20** (Original) (incorporating the rejection in claim 19), Liang discloses the system, wherein packaging system further registers the first OSGi bundle after being packaged with the first native application (e.g., Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12)

19. **As to claim 21** (Original) (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the communication system identifies the at least one prerequisite to the client device and receives a response from the client device that indicates whether the client device has the at least one prerequisite (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

20. **As to claim 22** (Original) (incorporating the rejection in claim 19), Klicnik discloses the system, further comprising: a prerequisite identification system for determining the at least one prerequisite; and an information generation system for generating the dependency information based on the at least one prerequisite (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

21. **As to claim 23** (Original) (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the at least one prerequisite comprises another native application (e.g., [0018], Lines 5-13)

22. **As to claim 24** (Original) (incorporating the rejection in claim 19), Liang discloses the system, wherein bundle loading system comprises:

- an export system for installing the first OSGi bundle and the second OSGi bundle within the OSGi environment of the client device (e.g., Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.);
- a deployment system for deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- a removal system for removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle

(e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

23. **As to claim 25** (Original) (incorporating the rejection in claim 19), Liang discloses the system, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (e.g., Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.)

24. **As to claim 26** (Original) (incorporating the rejection in claim 19), Klicnik discloses the system, wherein the client device includes: an analysis system for determining whether the client device has the at least one prerequisite; and a response system for generating and sending a response to the server (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

25. **As to claim 28** (Original) (incorporating the rejection in claim 27), Klicnik discloses the system, wherein the at least one prerequisite comprises another native application (e.g., [0018], Lines 5-13)

26. **As to claim 29** (Previously Presented) (incorporating the rejection in claim 19), Liang discloses the system, wherein the dependency information is expressed as a package import statement (e.g., Sec. of c) Let The Third Party Bundle To Manage the

Service Dependency In a Centralized Control Way, 3rd Para., Lines 5-11 – import-package field in its manifest file)

27. **As to claim 31** (Previously Presented), Klicnik discloses a program product stored on a recordable medium for resolving prerequisites for native applications, which when executed, comprises:

- program code for packaging a native application for a client device and corresponding dependency information within a first bundle on a server (e.g., [0010], Lines 10-16 – A bundle's manifest file identifies the bundle's contents and also the packages and services which are imported and exported by that bundle), wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (e.g., Fig. 2; [0031], Lines 9-24; Fig. 3 – all prerequisites items; [0032]);

Although Klicnik discloses OSGI bundles (e.g., [0010]), but does not explicitly disclose resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

However, in an analogous art of *Bundle Dependency in Open Services Gateway Initiative Framework Initialization*, Liang discloses resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (e.g., Abstract, Lines 1-11; Sec. 1 of Introduction, 1st Para., Lines 1-6; 2nd Para., Lines 1-12; Sec. of II Bundle Dependency During Framework Initialization, 1st Para., Lines 1-3; Fig. 3 –

bundle dependency relationship; Sec. of IV. Conclusions and Discussions, 1st Para., Lines 1-2 – some of the solutions provided here are constructed form the OSGi server side)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Klicnik's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Klicnik's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (e.g., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16)

Further, Klicnik and Liang do not explicitly disclose program code for polling the client device to determine if the client device has the at least one other prerequisite; program code for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and program code for loading the first bundle and the second bundle on the client device.

However, in an analogous art of *Runtime-Resource Management*, Clohessy discloses:

- program code for polling the client device to determine if the client device has the at least one other prerequisite (e.g., Fig. 4 – depicting loading one or

more new application component into a portable device - it shows the recursive path used to resolve prerequisites, steps 104 → 106 → 108 → 109 → 110 → 112 →104 etc. until step-114 or END reached; [0038], Lines 12-15 – Alternatively, **a server** could **identify** one or more new application components required to upgrade/update an application component already stored in **the portable device** (client device); [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed – emphasis added);

- program code for obtaining the at least one prerequisite (e.g., [0038], Lines 12-15 - ... one or more new application component required to upgrade/update an application component ...) if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle (e.g., [0035], Lines 7-10 – Preferably, each RDL is stored with its associated application component in a single file such as JAVA™ Archive (JAR) file or J9 executable (JXE) file that can be run by the IBM J9 VM) that is accessible to the server (e.g., [0041] – the client device determining its available resources, and these are communicated to the server since; [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed); and

- program code for loading the first bundle and the second bundle on the client device (e.g., Fig. 4, elements 108 and 114 show loading the final set of OSGi bundles on the client device)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Clohessy into the Klicnik-Liang's system to further provide program code for polling the client device to determine if the client device has the at least one other prerequisite; program code for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server; and program code for loading the first bundle and the second bundle on the client device in Klicnik-Liang's system.

The motivation is that it would further enhance the Klicnik-Liang's system by taking, advancing and/or incorporating Clohessy's system which offers significant advantages for improved resource management methods, systems, and products which protect runtime system resources from poorly designed or destructive application components as once suggested by Clohessy (e.g., [0006])

28. **As to claim 32** (Original) (incorporating the rejection in claim 31), Liang discloses the program product, wherein program code for packaging further registers the first OSGi bundle after being packaged with the first native application (e.g., Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 –

from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12)

29. **As to claim 33** (Original) (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the program code for polling identifies the at least one prerequisite to the client device and receives a response from the client device that indicates whether the client device has the at least one prerequisite (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

30. **As to claim 34** (Original) (incorporating the rejection in claim 31), Klicnik discloses the program product, further comprising: program code for determining the at least one prerequisite; and program code for generating the dependency information based on the at least one prerequisite (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

31. **As to claim 35** (Original) (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the at least one prerequisite comprises another native application (e.g., [0018], Lines 5-13)

32. **As to claim 36** (Original) (incorporating the rejection in claim 31), Liang discloses the program product, further comprising:

- program code for installing the first OSGi bundle and the second OSGi bundle within an OSGi environment of the client device (e.g., Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.);
- program code for deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- a removal system for removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

33. **As to claim 37** (Original) (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (e.g., [0018], Lines 5-13)

34. **As to claim 38** (Original) (incorporating the rejection in claim 31), Klicnik discloses the program product, wherein the client device includes: program code for determining whether the client device has the at least one prerequisite; and program code for generating and sending a response to the server (e.g., Figs. 4A-4C; Fig. 5; [0027]; [0043], Lines 1-3)

35. **As to claim 39** (Original) (incorporating the rejection in claim 31), please refer to claim **9** as set forth above accordingly.

36. Claims 8 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Klicnik in view of Liang, Clohessy and in further view of Yook et al., (Pub. No. US 2004/0139177 A1) (hereinafter 'Yook')

37. **As to claim 8** (incorporating the rejection in claim 1) (Previously Presented), Klicnik, Liang, and Clohessy do not explicitly disclose discloses the method wherein the method is performed recursively for the at least one prerequisite to resolve prerequisites for the at least one prerequisite.

However, in an analogous art of *System and Method for Managing Application*, Yook discloses the method wherein the method is performed recursively for the at least one prerequisite to resolve prerequisites for the at least one prerequisite (e.g., [0013] - ... a function of a controlled device can be dynamically extended by allowing a control device (e.g., the server) and the controlled device (e.g., client device) to control installation and management of the application and **continually updating the application** for use in the controlled device; Fig. 4 – illustrating an actual configuration of an application management system implemented in **an application server pull mode** – elements 413 – OSGi framework; 410 - Application server; 420 – Controlled device; Fig. 7 – **Application Server Pull Mode**; S720 – Extract positional information

on application file **from controlled device**; [0068], Lines 3-7 – Thus, the application management can be performed more efficiently by using either application server pull mode to be used for electric home appliances …; [0076] - ... **for continuously updating the application** ... – emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Yook into the Klicnik-Liang-Clohessy's system to further provide the method wherein the method is performed recursively for the at least one prerequisite to resolve prerequisites for the at least one prerequisite in Klicnik-Liang-Clohessy system.

The motivation is that it would further enhance the Klicnik-Liang-Clohessy's system by taking, advancing and/or incorporating Yook's system which offers significant advantages that the functions of the electric home appliances can be dynamically extended in the home network environment since an application management system operable independently of the home network middleware can be implemented as once suggested by Yook (e.g., [0075])

38. **As to claim 18** (Previously Presented) (incorporating the rejection in claim 11), please refer to claim **8** as set forth above accordingly.

39. Claims 10, 27, 30, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Klicnik in view of Liang, Clohessy and in further view of Hall et al.,

(*Component Deployment on OSGi: The Gravity Case, January 29, 2003, Fractal Workshop – LSR-Adele*) (hereinafter 'Hall')

40. **As to claim 10** (Original) (incorporating the rejection in claim 1), although Klicnik discloses OSGI bundles (e.g., [0010]) and Liang discloses bundle dependency during framework initialization (e.g., Sec. of II), but Klicnik, Liang, and Clohessy do not explicitly disclose the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

However, in an analogous art of *component deployment on OSGi: the gravity case*, Hall discloses the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle (e.g., Slide 7 – Bundle Manifest Example - Import-Package, specification-version)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hall into the Klicnik-Liang-Clohessy's system to further provide the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

The motivation is that it would enhance the Klicnik-Liang- Clohessy's system by taking, advancing and/or incorporating Hall's system which provides the framework of a factory service concept built on top of OSGi and further standardizes OSGi component

creation as once suggested by Hall (e.g., Slides 30-31, 35-41 – Extended OSGi Component Model for Gravity)

41. **As to claim 27** (Original) (incorporating the rejection in claim 19), although Klicnik discloses OSGI bundles (e.g., [0010]) and Liang discloses bundle dependency during framework initialization (e.g., Sec. of II), but Klicnik, Liang and Clohessy do not explicitly disclose the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application.

However, in an analogous art of *component deployment on OSGi: the gravity case*, Hall discloses the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application (e.g., Slide 7 – Bundle Manifest Example - Import-Package, specification-version)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hall into the Klicnik-Liang-Clohessy's system to further provide the system, wherein the dependency information specifies an identity and a version of the at least one prerequisite required by the native application.

The motivation is that it would enhance the Klicnik-Liang- Clohessy's system by taking, advancing and/or incorporating Hall's system which provides the framework of a factory service concept built on top of OSGi and further standardizes OSGi component creation as once suggested by Hall (e.g., Slides 30-31, 35-41 – Extended OSGi Component Model for Gravity)

42. **As to claim 30** (Original) (incorporating the rejection in claim 19), please refer to claim **10** as set forth above accordingly.

43. **As to claim 40** (Original) (incorporating the rejection in claim 31), please refer to claim **10** as set forth above accordingly.

Response to Arguments

44. Applicant's arguments filed on June 16, 2008 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

(**A.1**) The cited references fail to each or suggest obtaining the at least one prerequisite if the client device does not have the at least one prerequisite and loading the at least one prerequisite and the native application on the client device (see REMARKS on page 13, 1st paragraph; page 14, 1st paragraph)

(**A.2**) Yook shows no recurring events in FIG.7 or elsewhere (see REMARKS on page 15, 1st paragraph)

Examiner's response:

(**R.1**) Clohessy discloses that polling the client device by the server to determine if the client device has the at least one other prerequisite; obtaining the at least one prerequisite if the client device does not have the at least one prerequisite (e.g., Fig. 4 – depicting loading one or more new application component into a portable device - it shows the recursive path used to resolve prerequisites, steps 104 → 106 → 108 → 109 → 110 → 112 →104 etc. until step-114 or END reached; [0038], Lines 12-15 – Alternatively, **a server** could **identify** one or more new application components required to upgrade/update an application component already stored in **the portable device** (client device); [0042] – the server uses this information to determine if the client has sufficient resources before downloading an application component; by identifying the available resource on the client, any limitation is also disclosed – emphasis added) and loading the at least one prerequisite (e.g., [0038], Lines 12-15 - … one or more new application component required to upgrade/update an application component …) and the native application (e.g., [0035], Lines 7-10 – Preferably, each RDL is stored with its associated application component in a single file such as JAVA™ Archive (JAR) file or J9 executable (JXE) file that can be run by the IBM J9 VM) on the client device (e.g., [0046] – In the preferred embodiment, the new application component is **downloaded from a network server to the portable device** in a know manner)

(**R.2**) Yook teaches recurring events and a recursively process (e.g., [0013] - … a function of a controlled device can be dynamically extended by allowing a control device (e.g., the server) and the controlled device (e.g., client device) to control installation and management of the application and **continually updating the application** for use in

the controlled device; Fig. 4 – illustrating an actual configuration of an application management system implemented in **an application server pull mode** – elements 413 – OSGi framework; 410 - Application server; 420 – Controlled device; Fig. 7 – **Application Server Pull Mode**; S720 – Extract positional information on application file **from controlled device**; [0068], Lines 3-7 – Thus, the application management can be performed more efficiently by using either application server pull mode to be used for electric home appliances ...; [0076] - ... **for continuously updating the application** ... – emphasis added)

Conclusion

45. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192

/Eric B. Kiss/
Eric B. Kiss
Primary Examiner, Art Unit 2192

September 10, 2008